REMARKS

Status of Claims

Claims 1 – 26 were original in the application. Claims 4, 9, 17, and 22 have been cancelled. Claims 2, 5 – 8, 14 – 16 and 18 – 20 have been currently amended. Claims 1 – 3, 5 – 8, 10 – 16, 18 – 21 and 23 – 26 are submitted for examination on the merits.

Amendments to the Specification

Responsive amendments to the specification have been made in each instance.

Objections to the Claims

Responsive amendments to the claims have been made in each instance.

Rejection Pursuant to 35 USC 102

Claims 1 - 26 were rejected as being anticipated by Hellestrand et . al., U.S. Patent 6,230,114.

Regarding claim 1, the Examiner contends that Hellestrand discloses a virtual real time system for simulating a physical test environment comprising: A master computer module (the interface mechanism, column 14, lines 36-43); and at least one slave computer module communicated to the master computer module and having a clocked operation, which is synchronized to the master computer module (processor 1 simulator 208, FIG. 2); wherein the master computer module and at least one each slave computer module each have a launcher submodule (the kernel in the interface

mechanism, column 14, lines 36-43) and a deployment submodule (hardware simulator 203, FIG. 2), the launcher submodule for launching the deployment submodule and controlling the deployment submodule for synchronized operation with the master computer module, the deployment submodule generating a virtual clock and following commands from the launcher submodule (each processor simulator has its own concept of time, as does the hardware simulator, column 8, lines 51-57).

The Examiner has proposed that interface mechanism 119 of Fig. 2 is the same as the master computer module of claim 1. Hellestrand states at col. 4, lines 43 – 53:

> Described herein is a design system operating on a host computer system and simulating an electronic system that contains target digital circuitry and a target processor having a pipeline, the design system comprising a hardware simulator simulating the target digital circuitry, a processor simulator simulating the target processor executing a user program by executing the user program substantially on the host computer system, and **an interface mechanism that couples the hardware simulator with the processor simulator including passing information between the hardware simulator and the processor simulator**.

At col. 8, lines 17 – 47:

> **An interface mechanism 119** is coupled to both the processor simulator 107 and the hardware simulator 103 and **enables communication between processor simulator 107 and hardware simulator 103**. Processor simulator 107 includes a communication mechanism 141 to pass information to the hardware simulator 103 **using the interface mechanism** when an event requires interaction of user program 109 with the target digital circuitry. Such events include times when user program 109 encounters an input/output instruction, or when the program has an arithmetic exception during execution, and other significant events. The hardware simulator 103 also includes a communication mechanism 143 to pass information to processor simulator 107 **using the interface mechanism** at events significant to the hardware simulator 103 that need to be communicated to the processor simulator. Such an event includes when a signal in the target digital circuitry connected to the target processor is asserted, for example, an interrupt.
>
> **The interface mechanism 119 passes the information across the hardware/software boundaries.** The preferred embodiment interface mechanism 119 includes a message passing kernel. Thus, in the preferred embodiment, both the processor simulator and the hardware simulator communication mechanisms 141 and 143 are included in interface mechanism

> 119. Also, the processor simulator and the hardware simulator are tasks under the kernel, and the kernel provides the mechanism for the tasks to communicate whenever one or the other task requires it. When several processor simulators operate, each runs independently as a task under the kernel.

In contrast the master computer module of claim 1 is not simply a means to enable communication between a processor simulator and a hardware simulator as is the case for the interface mechanism 119. As stated in paragraph [0019] the master computer module includes a launcher submodule and a deployment submodule. The launcher submodule launches the deployment submodule and controls the deployment submodule for synchronized operation. The deployment submodule generates a virtual clock and follows the commands from the launcher submodule. As stated in paragraph [0020] the launcher submodule in the master computer module is a central virtual real time controller for the system. As stated in paragraph [0021] the deployment submodule in the master computer module generates a virtual clock signal based on process CPU instruction execution. It is clear that the interface mechanism 119 of Hellestrand is a communications module for passing information between the processor simulator and hardware simulator. The master computer module is the controller which allows a set of programs to execute together using virtual real-time behavior as though they were executing in real-time on target hardware. However, under the control of master computer module the programs execute like normal programs (i.e. in virtual or process time) insofar as an outside observer (i.e. the operator) is concerned. The programs can therefore be executed on a normal timeshared workstation where precise time allocation and coordination is not provided and performance is more, less, or varying with respect to the target hardware, yet the programs interact with each other as though they were executing in real-time on the target hardware. The functional

relationship of the target hardware and processor is preserved whether or not the execution is actually running in real time or not. This allows a faithful simulation to be run in unmodified workstations at slower or different timings that exist in reality. Interface mechanism 119 does not perform this function.

Regarding claim 2, the Examiner contends that Hellestrand discloses a launcher submodule in the master computer module which is a central virtual real time controller for the system (simulation time is started and maintained by the hardware simulator, column 9, lines 58-63).

Hellestrand at Col. 9, lines 58-63 state:

> In the preferred embodiment, the hardware simulator provides a simulation time frame for the design system. That is, simulation time is started and maintained by the hardware simulator, and whenever synchronization is required, **all times are synchronized to the hardware simulation time**, which is the simulation time for the system.

It is clear that in Hellestrand the entire system is synchronized to the **hardware simulation time**. In contrast as stated at paragraph [0043] for example in the claimed invention, each slave program, e.g. Flight, Simulation and Ground software, has two executable programs, called a launcher and a deployment. A launcher's job is to launch its deployment and provide the necessary controls for VRT synchronization. A deployment is where the software program resides. Virtual clock ticks are generated from the deployment, and launcher controls it via VRT commands. The actual software program resides in the deployment. As an example Flight software program running as a Unix process would be a master deployment and Simulation software program running as a Unix process would be slave deployment. There would be a separate launcher program for Flight software--called a master launcher, and a separate launcher

program for Simulation software--called a slave launcher. Each slave generates its own virtual clock. As stated in paragraph [0045] the virtual real time normally starts from 0 and increments with VRT ticks. *VRT clock speed can be adjusted or scaled up or down using setScaleFactor command.* The setTickResolution allows users to adjust the virtual clock synchronizing resolution among the various software deployments. Virtual clock resolution is dependent on the operating system and the workstation hardware. The claimed system is coordinated to a virtual real time controller dependent on the operating system environment for which there is no equivalent in Hellestrand, which instead is temporally synchronized to the hardware simulator which seeks to simulate real hardware time.

Regarding claim 3, the Examiner contends that Hellestrand discloses the deployment submodule in the master computer module generates a virtual clock signal based on process CPU instruction execution (The hardware simulator provides the simulation time frame, column 19, lines 13-15).

Hellestrand at col. 19, lines 13 – 15 states:

> The hardware simulator provides the simulation time frame. Any units of time may be used, and clock cycles will be assumed to be the unit of time.

Hellestrand refers to clock cycles in the hardware simulator as a unit of time. Claim 3 calls for the deployment submodule in the master computer module to generate a virtual clock signal based on process CPU instruction execution. Process CPU instruction execution could take more than one clock cycle and usually do. VRT timing is thus also dependent on the process being executed and its instruction executions instead of a chip clock.

Regarding claim 5, the Examiner contends that Hellestrand discloses comprising

a test master computer submodule communicating with the master launcher submodule for configuring the system and advancing, starting, stopping, adjusting and monitoring virtual real time, and/or issuing time related commands to the deployment submodule in the master computer module (the simulation is run under debugger control, column 15, line 64, through column 16, line 2).

Hellestrand states at column 15, line 64, through column 16, line 2:

> Prior to execution, the user may insert debugger breakpoints in the user programs for each processor simulator. Prior to execution the user can enable or disable the breakpoints. As the simulation is run under debugger control, whenever a breakpoint is encountered, the debugger stops execution.

Fixed debugging software installed to provide stop or break points does not comprise a test master computer submodule to configure the system and advance, start, stop, adjust and monitor the virtual real time, and issue time related commands to the deployment submodule in the master computer module.

Regarding claim 6, the Examiner contends that Hellestrand discloses the master deployment submodule generates a virtual clock signal and where test master computer submodule generates scale-up and/or scale-down commands of the virtual clock in the master deployment submodule (Any units of time may be used, column 19, lines 13-1 5).

As argued in regard to the prior claims, Hellestrand does not disclose a master deployment submodule which generates a scalable virtual clock signal. An arbitrary choice of time units in the hardware simulator of Hellestrand does not mean that the clock signal is either virtual or scalable during operation. Hellestrand does not disclose scale commands.

Regarding claim 7, the Examiner contends that Hellestrand discloses the slave

launcher submodule comprises a slave launcher synch submodule which, upon receiving a command from the master launcher submodule (a start signal in the digital circuitry starts the processor simulator for processor 1, column 19, lines 15-26), requests the corresponding slave deployment submodule via the slave launcher synch submodule to advance the slave deployment submodule by a predetermined number of virtual clock ticks and to stop (Processor 1 executes for a time AT2 until T3, column 19, lines 15-26), after which the slave deployment submodule suspends operation and waits for the slave launcher submodule to resume operation (to suspend operation of processor simulator 207, column 19, lines 15-26).

Hellestrand describes an operation where processor 1 simulator 208 operates until an event, executes a function and then send a stop command to a second processor 2 simulator 207 to stop. Issuance of a stop command to a second simulator has virtually no relevance to advancing the virtual clock in a slave deployment submodule and then waiting for a slave launcher submodule to resume operation of the slave deployment submodule. There is no virtual clock manipulation in Hellestrand.

Regarding claim 8, the Examiner contends that Hellestrand discloses that the master launcher submodule sends a start-tick command to only to the slave launcher submodule, if it is prepared to receive the next start-tick command by sending a socket call with a start-tick message (a start signal in the digital circuitry starts the processor simulator for processor 1, column 19, lines 15-26).

As disclosed at paragraph [0047] FIG. 4 provides a detailed flow chart of the master launcher 304, which controls and synchronizes the whole system on VRT ticks. The initial parameters such as slave deployment tick synchronize size, number of ticks

to advance, assign id to each slave, tick resolution setup and scale factor setup is done in step 401. Master launcher 304 sends a start-tick command to only those slave launchers 307, which are prepared to receive the next start-tick command (steps 403, 404 and 405) i.e. where enough time has been allowed to synchronize with the master 304. A socket call with start-tick message is sent at step 405 to the slave launcher 307. The socket call helps the VRT to support distributed computing. Again referring to the disclosure of Hellestrand above, there is no relevance to the sending of a stop command between two simulators and the virtual clock commands using a socket call.

Regarding claim 10, the Examiner contends that Hellestrand discloses that the slave launcher submodule after receiving a start-tick command from the master deployment submodule sends a SIGCONT signal to the suspended slave deployment submodule (a start signal in the digital circuitry starts the processor simulator for processor 1, column 19, lines 15-26), the slave launcher submodule sends an acknowledgment message to the master launcher submodule, the slave deployment submodule in parallel with other programs runs the requested number of ticks (Processor 1 executes for a time AT2 until T3, column 19, lines 15-26).

The referenced sections of Hellestrand, which are also discussed above, do not in any way related to a protocol where a slave deployment submodule runs a requested number of ticks of a virtual clock. Clock manipulation signals among system submodules is not disclosed in Hellestrand.

Regarding claim 11, the Examiner contends that Hellestrand discloses that the master launcher submodule then sends a signal SIGCONT to its corresponding master deployment submodule to run a requested number of virtual clock ticks based on Vclk

clock ticks which are generated . when the time consumed by execution of process CPU instructions is equal to or greater than tick-resolution time, the master deployment submodule suspends its operation after running the requested number and the master launcher submodule waits for the master deployment submodule to complete its cycles (The waitEvent function waits for the occurrence of any event or time out on the given delay, column 17, lines 35-40).

Hellestrand discloses a interface command, waitEvent, which causes operation to wait for an event or timeout on a given delay time and then the elapsed delay time is returned by the command. This is not a command function which sends a signal to its corresponding master deployment submodule to run a requested number of virtual clock ticks based on Vclk clock ticks which are generated when the time consumed by execution of process CPU instructions is equal to or greater than tick-resolution time. Such a command is exactly the opposite of a wait or operation suspension. The master deployment submodule suspends its operation *only after running* the requested number of ticks.

Regarding claim 12, the Examiner contends that Hellestrand discloses that the master launcher submodule sends a stop-tick message to each slave launcher submodule which needs to be synchronized at that clock tick based on slave tick synchronize size and a stop-tick socket call is made to the candidate slave launcher submodule (a library of functions is provided . . . to affect synchronizations, column 16, lines 18-35).

The section of Hellestrand cited relates to a library of functions which allow communication between differing data formats of the process simulator and hardware

simulator. Such communications are recited as "affecting synchronization", but are not shown to be synchronization commands as such. In claim 12 as disclosed at paragraph [0050] master launcher 304 sends a stop-tick message to each slave launcher 307 at steps 409 and 412 of Fig. 4 that is needed to be synchronized at that tick based on slave tick synchronize size. A tick synchronize size is a smallest time step at which a slave program could synchronized with the master program with violating the causality effects between the two programs. A tick synchronize size is determined by the system designer.

Regarding claim 13, the Examiner contends that Hellestrand discloses that the slave launcher submodule after receiving a stop-tick command waits for a SIGSTOP signal from the slave deployment submodule to make sure that the requested number of virtual clock ticks has been completed, and the slave launcher submodule sends a stop-tick acknowledgment message to the master launcher submodule (The asynchronous event handler function is called when an asynchronous event occurs, column 18, lines 25-39).

The referenced section of Hellestrand refers in general to the handling a asynchronous events. In claim 13 a protocol is defined by synchronization with a virtual clock tick can be maintained.

Regarding claims 14-26, the Examiner contends that Hellestrand discloses that these method claims include equivalent limitations as in claims 1 - 13 and are anticipated using the same analysis of claims 1 - 13.

Claims 14 – 26 have been amended where appropriate to correspond to the amendments of claims 1 - 13 and the grounds of distinction of the various claims as set

forth above re reincorporated.

Applicant respectfully requests advancement of the claims to allowance.

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on January 5, 2007 by

Signature

January 5, 2007

Respectfully submitted,

Daniel L. Dawes
Registration No. 27,123
Myers Dawes Andras & Sherman LLP
19900 MacArthur Blvd., 11th Floor
Irvine, CA 92612
(949) 223-9600